1. Notes for presentation

The Common Weakness Enumeration is a category system for hardware and software weaknesses and vulnerabilities. It is currently maintained by Mitre but you may find other maintainers as well. This enumeration provides a common way of describing weaknesses. This kind of list allows for better understanding and collaboration between developers.

As my project will be mainly focusing on open source vulnerability scanning software for web applications I will be focusing on these 3 CWEs (I will also mention them in a minute

- CWE-89: Improper Neutralization of Special Elements used in an SQL Command
- CWE-552: Files or Directories Accessible to External Parties
- CWE-1104: Use of Unmaintained Third Party Components

Project will later contain introduction to open source web vulnerability scanners and how they operate. This section should give reader a better understanding on their functionality and technical details.

Next, the relevance of open source tools in web security will be explained as I belive this to be key part in my project. Arguably, Open source has never been more relevant and with IT moving so fast the security has never been a bigger topic. Not only does open source help starting developers but also professionals and according to a report by Red Hat, 95% of IT leaders agree that open-source solutions are strategically important to their organization'‘s overall enterprise infrastructure software strategy. – for later **https://www.redhat.com/en/enterprise-open-source-report/2022**

Focus Areas of my project will be on CWEs mentioned before. SQL injection, hidden file detection and outdated component detection. Requeirements for choosing these will be mentioned in a minute

Importance of controlled testing evnironment is a part which I belive to be important to mention. Tests not executed under controlled environment may lead to unexpected behavior and wrong test results. This section will have a concise description of my environment and testing applications.

next CWEs next

Requirements for choosing tools include popularity and toolset. To test CWEs selected for this project the tools need to have the appropriate functionality. As there are many applications which tackle these problems I decided to choose by popularity. To make experiments more relevant I chose an additional app which specializes in only in finding hidden files that is dirsearch. When selecting the tools, I considered several important criteria: Coverage – whether the tool can detect issues related to the CWEs I'm testing, such as SQL injection or outdated components. Popularity & Community Support – I focused on tools with active communities or frequent updates, since these are more likely to stay effective against modern vulnerabilities. Reporting Features – I also looked at how well the tool presents its results — some output detailed HTML reports, others just console logs. Open-source license – because my project focuses on open-source tools, licensing was also a factor.

next

w3af (Web Application Attack and Audit Framework) Python-based framework developed by OWASP. Very extensible — it supports plugins for scanning and exploiting vulnerabilities like SQLi, XSS, and more. Has both GUI and command-line interfaces. It's considered a "framework" rather than just a scanner because it allows chaining multiple plugins for different attack types. Strength: powerful and customizable. Nikto One of the oldest and most popular web vulnerability scanners. Checks for over 6,000 potential issues like outdated software versions, misconfigurations, and dangerous files. Simple command-line usage, but reports can be verbose. Strength: very broad vulnerability coverage.

Wapiti Python-based, active project with good modern support. Strength: good balance of speed and detection accuracy.

Vega GUI-based vulnerability scanner written in Java. More visual and beginner-friendly than CLI tools. Has built-in modules for cwes chosen

Grabber Lightweight scanner — ideal for smaller sites or development testing. Supports detection of SQLi, XSS, file inclusion, and JavaScript source code analysis. Very fast because it's minimalistic.

Dirsearch Specialized tool — focuses only on directory and file discovery (CWE-552). Uses brute-force scanning to identify hidden files or backup directories that are publicly accessible. Often used alongside general scanners to complement their results. Strength: extremely fast and efficient.

next

Comparison Between General-Purpose and Specialized Tools Now that I've covered the individual tools, it's important to note that they fall into two wider categories: General-purpose scanners, like w3af, Nikto, Wapiti, Vega, and Grabber. Specialized tools, like Dirsearch.

General-purpose scanners aim to find as many types of vulnerabilities as possible. They are versatile and can detect SQL injection, outdated components, and common misconfigurations. However, they may miss hidden files or very specific vulnerabilities because they spread their detection over many areas. Specialized tools, on the other hand, focus on one job and do it extremely well. Dirsearch, for example, doesn't care about SQL injection or cross-site scripting — it's purely designed to brute-force directories. This makes it faster and more accurate for that specific use case. In my project, I'll use both types — general-purpose scanners for overall coverage, and specialized tools like Dirsearch for deeper, more focused testing. This approach allows me to cross-verify findings and increase reliability of results.

next

Tests will take place in environment described in one of the previous sections and they will be done in 2 waves. First tests are conducted with default settings. Second wave of tests will be done with optimal configuration. Emphasis will be put on the tests with optimal configuration as those results will have highest relevance.

Testing will be conducted on these 3 targets: Custom vulnerable web application laravel application (Damn Vulnerable Web Application)